**Подготовка Sprite Sheet Run**

Продолжаем дело, начатое в первой части. Сейчас у нас есть платформа и стоящий на ней персонаж с анимацией покоя. Настало время научить нашего персонажа бегать вправо-влево по платформе.

Загрузим сцену из первой части. Напомню, что в прошлый раз мы импортировали несколько спрайтов в папку *Assets*— *Sprites*. Среди них должен быть спрайт под названием *Run*. Мы будем использовать его для создания анимации бега. Для этого нам надо проделать те же действия по превращению одиночного спрайта в коллекцию, как и при создании анимации покоя. Вкратце напомню: выделяем спрайт, в окне *Inspector*устанавливаем свойство *Sprite Mode* как *Multiple*, нажимаем ниже *Sprite Editor*, нарезаем изображение в режиме *Grid*или *Automatic*.

**Добавляем анимацию Run**  
теперь в окне *Hierarchy*выбираем *Character*и переходим в окно *Animation*. Нажимаем на поле с анимацией *Idle*и выбираем *Create New Clip*, чтобы создать анимацию бега. Сохраним файл анимации в папке *Assets*— *Animations*под именем *Run*.

Новая созданная анимация *Run*стала текущей в окне *Animation*. Разворачиваем спрайт *Run*в окне *Project*, выделяем все фалы *Run\_0… Run\_9* и перетаскиваем в окно *Animation*. Установим пока значение *Sample*равное 24.

**Создаем переходы в Animator**  
все это мы уже делали в первой части, а теперь будет нечто новое. Перейдем в окно *Animator*. Сейчас там отображены три анимации: *Any State*, *Idle*и *Run*. Нам предстоит задать условия перехода из анимации *Idle*в анимацию *Run*, то есть из состояния покоя в состояние бега. В нижнем левом углу есть поле *Parameters*. Нажимаем на плюсик, выбираем *Float*и называем новый параметр как *Speed*. Тем самым мы создали параметр типа число с плавающей запятой, обозначающий скорость перемещения персонажа. Именно в зависимости от значения этого параметра будет происходить переключение из анимации покоя в анимацию бега. Теперь нажимаем правой кнопкой мыши на анимацию *Idle*, выбираем *Make Transition* и нажимаем левой кнопкой мыши на анимацию *Run*. Между анимациями появится линия со стрелкой. Передвиньте мышкой прямоугольники анимации, если плохо видно. Кликнем по линии со стрелкой. В окне *Inspector*отобразятся свойства перехода между анимациями. Обратим внимание на низ окна, в раздел *Conditions*. Кликнем на параметр *Exit Time* и поменяем его на *Speed*. Второе поле *Greater*оставим без изменений, а в третьем введем значение*0.01*. Мы создали условие перехода из анимации покоя в анимацию бега — оно происходит, когда значение параметра скорости становится немногим больше нуля.  
  
Теперь нужно сделать обратный переход — из *Run*в *Idle*. Делаем все с точностью наоборот: *Make Transition* от *Run*к *Idle*, выделяем переход, в *Conditions*устанавливаем *Speed*— *Less*— *0.01*.

**Пишем скрипт движения**  
  
теперь у нас есть две анимации и условия перехода между ними. Но пока ничего работать не будет, потому что все что мы сделали нужно «оживить» при помощи скрипта. Давайте перейдем в окно *Project*и создадим в папке *Assets*подпапку *Scripts*. Добавим в нее новый *C# Script*, назовем его *CharacterControllerScript*и откроем на редактирование.  
  
Я приведу полный листинг скрипта с подробными комментариями, а ниже еще поясню, что в нем происходит.

**using** UnityEngine;

**using** System.Collections;

**public** **class** **CharacterControllerScript** : **MonoBehaviour**

{

*//переменная для установки макс. скорости персонажа*

**public** **float** maxSpeed = 10f;

*//переменная для определения направления персонажа вправо/влево*

**Private** **bool** isFacingRight = **true**;

*//ссылка на компонент анимаций*

**private** Animator anim;

///<summary>

/// *Начальная инициализация*

///</summary>

**private** **void** **Start**()

{

anim = GetComponent<Animator>();

}

///<summary>

/// *Выполняем действия в методе FixedUpdate, т. к. в компоненте Animator персонажа*

/// *выставлено значение Animate Physics = true и анимация синхронизируется с расчетами физики*

///</summary>

**private** **void** **FixedUpdate**()

{

*//используем Input.GetAxis для оси Х. метод возвращает значение оси в пределах от -1 до 1.*

*//при стандартных настройках проекта*

*//-1 возвращается при нажатии на клавиатуре стрелки влево (или клавиши А),*

*//1 возвращается при нажатии на клавиатуре стрелки вправо (или клавиши D)*

**float** move = Input.GetAxis("Horizontal");

*//в компоненте анимаций изменяем значение параметра Speed на значение оси Х.*

*//приэтом нам нужен модуль значения*

anim.SetFloat("Speed", Mathf.Abs(move));

*//обращаемся к компоненту персонажа RigidBody2D. задаем ему скорость по оси Х,*

*//равную значению оси Х умноженное на значение макс. скорости*

rigidbody2D.velocity = **new** Vector2(move \* maxSpeed, rigidbody2D.velocity.y);

*//если нажали клавишу для перемещения вправо, а персонаж направлен влево*

**if**(move > 0 && !isFacingRight)

*//отражаем персонажа вправо*

Flip();

*//обратная ситуация. отражаем персонажа влево*

**else** **if** (move < 0 && isFacingRight)

**Flip**();

}

///<summary>

/// *Метод для смены направления движения персонажа и его зеркального отражения*

///</summary>

**private** **void** **Flip**()

{

*//меняем направление движения персонажа*

isFacingRight = !isFacingRight;

*//получаем размеры персонажа*

Vector3 theScale = transform.localScale;

*//зеркально отражаем персонажа по оси Х*

theScale.x \*= -1;

*//задаем новый размер персонажа, равный старому, но зеркально отраженный*

transform.localScale = theScale;

}

}

**Последний штрих**  
Итак, мы завели несколько переменных: для задания максимальной скорости перемещения, для определения направления (вправо/влево) и для работы с компонентом *Animator*. Почти все действия происходят в методе *FixedUpdate*. В нем мы получаем значение оси *Х*, которое меняется при нажатии на клавиатуре клавиш влево-вправо или A-D (если не меняли соответствующие настройки проекта!). Затем устанавливаем это значение параметру *Speed*компонента *Animator*. Обратите внимание, что мы берем модуль этого значения при помощи метода*Mathf.Abs*, так как при создании условий перехода между анимациями покоя и бега мы сравниваем значение параметра с положительным числом *0.01*. Нам здесь не важно, в какую сторону бежит персонаж. Важно лишь величина значения. Далее задаем скорость перемещения по оси *Х* в соответствии со значением максимальной скорости. И, наконец, проверяем, в какую сторону бежит персонаж, и в какую сторону он в этот момент повернут. Если он бежит вправо, а повернут влево — разворачиваем его вправо путем инвертирования его размера по оси *Х*. И наоборот. Этим нехитрым способом мы избавились от необходимости делать две анимации вместо одной: для бега вправо и для бега влево.

Сохраняем скрипт. В Unity перетаскиваем его на нашего *Character*в окне *Hierarchy*. Запускаем игру, нажимаем влево-вправо или A-D.

Персонаж теперь умеет бегать! Скорость анимации получилась быстроватой. Ее можно снизить путем уменьшения значения *Sample*в окне *Animation* для анимации *Run*(значение 12 будет нормально). Если одновременно с игрой у вас видно окно *Animator*, то вы увидите, что во время покоя работает анимация *Idle*(бегает синий прогрессбар), а во время бега происходит переход на анимацию *Run*, и, соответственно, работает она.